VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Program pro testování přípravku PPZ082

Program for testing the product PPZ082

2012 Petr Gašparovič

Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Gašparovič**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Program pro testování přípravku PPZ082
Program for Testing the Product PPZ082

Zásady pro vypracování:

Cílem práce je implementace programového nástroje určeného pro testování přípravku PPZ082. Program musí provést testování přípravku PPZ082, pomocí etalonu PPZ08206 na všech pozicích.

1. Seznamte se s přípravkem PPZ082.
2. Seznamte se s komunikačním protokolem MODBUS.
3. Naprogramujte modul pro komunikaci protokolem MODBUS.
4. Vytvořte program pro kontrolu přípravku.
5. Zpracujte kompletní dokumentaci.

Seznam doporučené odborné literatury:

The Modbus [online]. 2011 [cit. 2011-11-22]. The Modbus. Dostupné z WWW: <http://modbus.org/>.
Datové listy přípravku PPZ082

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Věříš, Ph.D.**

Konzultant bakalářské práce: Ing. Martin Milata

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA FAKULTA ELEKTROTECHNIKY A INFORMATIKY

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení Studenta

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Pardubicích dne 3.5.2012 .....

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava, s výjimkou odstavce 3.1."

V Pardubicích dne 3.5.2012 .....

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Janu Věříši, Ph.D. za účinnou metodickou, a odbornou pomoc a další cenné rady při zpracování mé práce. Dále Ing. Martinu Milatovi za pedagogickou pomoc.

Abstrakt

Cílem této bakalářské práce je vytvoření aplikace, která pomůže usnadnit kontrolu přípravků pro nastavení tlakových snímačů ve firmě Elgas s.r.o. V teoretické části je přiblížení protokolu Modbus, pomocí něhož komunikuje měřící procesor těchto přípravků. Je zde popsána komunikace a funkce protokolu, způsob přenosu zpráv či způsob zpracování chybových stavů. Dále je v této části seznámení s přípravkem PPZ082 a potřebnými funkcemi pro zpracování požadavků. A na konec je probrán obvod a knihovna FTDI, který zprostředkovává komunikaci přípravku přes USB rozhraní. Praktickou část tvoří popis implementace aplikace, návrh struktury a rozbor jednotlivých metod s ukázkami kódu.

Abstract

The aim of this thesis is to create an application that will help to facilitate preparations for setting the control pressure sensors in the company Elgas Ltd. The theoretical part is an approximation Modbus protocol, which communicates with the processor measurement products. It describes the communication protocol and functions, method of transmission of messages and how to handle error conditions. Further, the introduction to this section with PPZ082 and functions necessary for processing requests. And at the end of the circuit is discussed and the FTDI library that mediates communication of USB interface. The practical part consists of a description of the implementation of applications, design and structure analysis of each method with examples of code.

Klíčová slova

Převodník, PPZ082, MODBUS, sériová linka, funkce, FTDI, rozhraní, rámec zprávy

Keywords

converter,PPZ082, MODBUS, serial link, funkction, FTDI, interface, frame message

Seznam použitých symbolů a zkratek

Zkratka	Popis
API	Application programming interface
A/D	Analog/Digital
ADU	Application Data Unit
ASCII	American Standard Code for Information Interchange
CRC	Cyclic Redundancy Check
EEPROM	Electrically Erasable ProgrammableRead-Only Memory
FTDI	Future Technology Devices International
GPIO	General Purpose Input/Output
LRC	Longitudinal Redundancy Check
LSB	Least Significant Bit
MSB	Most Significant Bit
PDU	Protocol Data Unit
PPZ082	Přípravek a pomocné zařízení 082
PID	ID produktu
RTU	Remote Terminal Unit
TCP/IP	Transmission Control Protocol/Internet protocol
UART	Universal asynchronous Receiver/transmitter
USB	Universal serial bus
VCP	Virtual COM port
VID	Vendor ID
MBAP	Modbus Application Header

Obsah

1. Úvod

Hlavním cílem této bakalářské práce je vytvoření aplikace, která bude s pomocí obsluhy testovat přípravky pro nastavení tlakových převodníků PPZ082 ve firmě ELGAS s.r.o, kde jsem zaměstnán. Tato firma se zabývá výrobou velice přesných plynoměrů a tlakový převodník, je jedna z jeho hlavních částí. Při nastavování těchto tlakových převodníků je velice důležité, aby přípravek na kterém nastavení probíhá, byl zcela v pořádku. Proto byla zavedena povinná kontrola, při které je potřeba tyto přípravky kontrolovat. Tato aplikace by měla přispět k ulehčení této pravidelné kontroly přípravků, jejichž počet se v současné době s trojnásobil z důvodu nárůstu výroby převodníků a celkové výroby přepočítávačů. Pracovník, který bude tyto kontroly provádět, by měl po přečtení přiloženého manuálu, bez problémů testovat přípravky s uživatelskou znalostí PC. Program byl vytvořen za pomoci vývojového nástroje Microsoft Visual Studia 2010 Professional, v objektově orientovaném programovacím jazyce C#. Bakalářská práce je rozdělena na čtyři okruhy . Nejprve je seznámení s protokolem Modbus, principy zasílání a příjmu zpráv a jeho datový model. Druhým okruhem je samotný popis přípravku PPZ082 a funkcí, které je potřeba naimplementovat. V další části práce se seznámíme s knihovnou FTDI, pomocí níž je zajištěn převod USB na sériové rozhraní a komunikace s ním. A ve čtvré části je popis samotné implementace aplikace.

2. Protokol Modbus

2.1. Seznámení

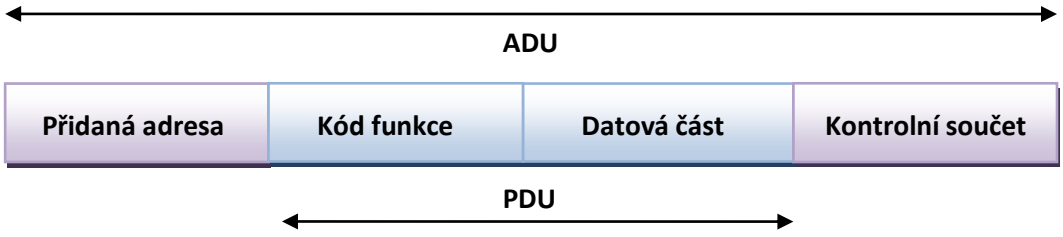
MODBUS byl vytvořen firmou MODICON. Je to komunikační protokol. Pracuje na úrovni aplikační vrstvy ISO/OSI modelu, a vytváří komunikaci typu klient-server mezi zařízeními na různých typech sítí. Komunikace tedy probíhá formou požadavek-odpověď. Modbus podporuje řadu komunikačních médií – sériové linky RS-232, RS-422 , optické a rádiové sítě nebo síť Ethernet s využitím protokolu TCP/IP.Požadované funkce jsou specifikovány kódem funkce, jež je součástí požadavku.

2.2. Popis

Struktura zprávy je na úrovni protokolu – PDU nezávislá na typu komunikační vrstvy. Podle typu sítě, kde je protokol použitý, je PDU rozšířena o další části a vytvoří tak zprávu na aplikační úrovni – ADU.

Max. velikost PDU na sériové lince:
256 bajtů
– adresa serveru (1 bajt)
– kontrolní součet CRC(2 bajty)
= 253 bajtů.
Potom:
Velikost ADU na RS-485:
253 bajtů PDU
+ adresa (1 bajt)
+ CRC (2 bajty)
= 256 bajtů
Velikost ADU na TCP/IP :
253 bajtů PDU + MBAP
= 260 bajtů

Maximální velikost PDU je zděděna z první implementace MODBUSu na sériové lince RS-485(256bajtů), kde tomu odpovídá maximální velikost PDU 253 bajtů.



Obr. 1: Tvar Modbus zprávy

Jednotlivé části Modbus zprávy jsou zobrazeny na obrázku 1. Operace, které chceme provést, rozlišujeme pomocí funkčního kódu Modbus zprávy , který v některých případech obsahuje i kód podfunkce upřesňující blíže funkci. Rozsah kódů je 1 až 255, kódy 128 až 255 jsou vyhrazeny pro záporné odpovědi - chyby. Operace, které chceme provést, rozlišujeme pomocí funkčního kódu, někdy obsahují i kód podfunkce upřesňující blíže funkci. Data předané ve zprávě slouží k provedení zadané funkce.
Můžeme předávat například adresu, počet vstupů či registrů, které má server přečíst nebo zapsat. Někdy nejsou další data potřeba, potom datová část ve zprávě chybí.

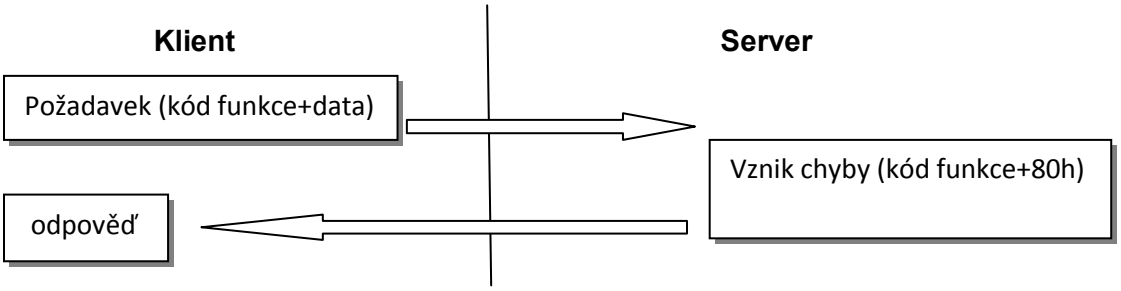
Protokol MODBUS definuje 3 základní typy zpráv (PDU):

-Požadavek (Request PDU)
- Kód funkce (1bajt)
- Datová část požadavku (n bajtů)

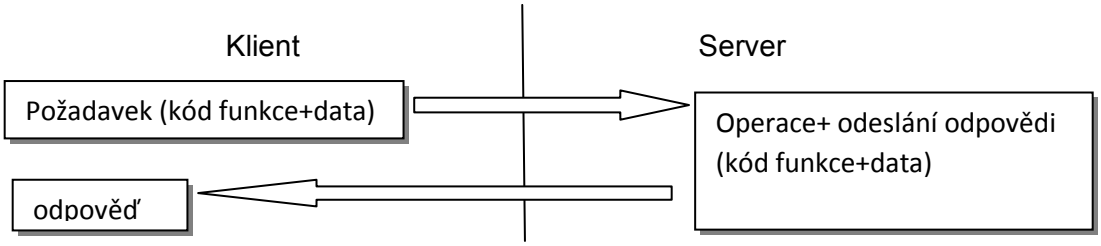
-Odpověď (Response PDU)
- Kód funkce (kopie z požadavku-1 bajt)
- Datová část odpovědi (n bajtů)

-Záporná odpověď (Exception Response PDU)
- Kód funkce + 80h (1 bajt)
- Chybový kód (identifikace chyby-1 bajt)

Pokud při provádění operace nedojde k chybě, odpoví server zprávou, která v poli funkce vrací kód provedené (požadované) funkce jako úspěšného vykonání požadavku. V datové části odpovědi jsou požadovaná data vrácená funkcí (Obr. 3). Pokud při vykonávání funkce dojde k chybě, je v poli funkce vrácen kód požadované funkce s nastaveným nejvyšším bitem-indikujícím neúspěch (exception response). V datové části je kód upřesňující důvod neúspěchu (Obr. 2).



Obr. 2: MODBUS transakce s chybou při provádění požadavku



Obr. 3: MODBUS transakce s bezchybným provedením požadavku

2.3. Typy funkcí

MODBUS definuje tři skupiny funkcí:

Veřejné kódy funkcí (Tab. 1)

- jsou unikátní
- schvalovány společností MODBUS-IDA.org
- veřejně zdokumentované
- zahrnují veřejné kódy funkcí i nepřiřazené kódy rezervované pro budoucí použití

Uživatelsky definované kódy funkcí

- uživatelsky definované kódy funkcí v rozsahu: 65 ÷ 72 a 100 ÷ 110
- umožňují uživateli vlastní implementaci funkce, která není definována
- není garance unikátnosti kódů

Rezervované kódy funkcí

-kódy funkcí používané některými firmami a nedostupné pro veřejné použití

Seznam veřejných kódů funkcí Modbus :

Specifikace				Kód	Podfunkce	hex
Přístup k datům	Bitový přístup	Fyzické diskrétní vstupy Interní bity nebo fyzické cívky	Čti diskrétní vstupy	02		02
			Čti cívky	01		01
			Zapiš jednu cívku	05		05
			Zapiš více cívek	15		0F
	16 bitový přístup		Čti vstupní registr	04		04
			Čti uchovávající registry	03		03
			Zapiš jeden registr	06		06
			Zapiš více registrů	16		10
			Čti/zapiš více registrů	23		17
			Zapiš registr s maskováním	22		16
			Čti FIFO frontu	24		18
	Přístup k záznamům v souborech		Čti záznam ze souboru	20	6	14
			Zapiš záznam do souboru	21	6	15
Diagnostika			Čti stav	07		07
			Diagnostika	08	00-18,20	08
			Čti čítač kom. událostí	11		0B
			Čti záznam kom. událostí	12		0C
			Sděl identifikaci	17		11
			Čti identifikaci zařízení	43	14	2B
Ostatní			Zapouzdřený přenos	43	13,14	2B
			CANOpen základní odkaz	43	13	2B

Tabulka č.1: Modbus funkce

2.4 Režimy Modbus protokolu

MODBUS standard definuje kromě aplikační vrstvy ISO/OSI modelu i některé implementace protokolu na konkrétní typ sítě nebo sběrnice. Příkladem je MODBUS na TCP/IP a MODBUS na sériové lince. Nás zajímá sériová linka, kde jsou definovány dva režimy - MODBUS RTU a MODBUS ASCII. Režim určuje formát vysílaných dat a jejich dekódování. Jednotky musí pracovat ve stejném vysílacím režimu.

Modbus ASCII

Režim ASCII je nepovinný. Každý 8 - bitový bajt je posílán jako 2 znaky ASCII (př. : hexadecimální hodnota “3E” je poslána jako dva znaky ASCII “3E“). Tento režim umožňuje posílání znaků až s 1 sekundovými mezerami. Začátek zprávy je znak „:“ a její konec řídícími znaky CR, LF. V tomto módu se k detekci chyb využívá LRC pole. Přenos je možný i bez použití parity, která je nahrazena druhým stop bitem. Rozklad zprávy na jednotlivé části, je na obr. 4.

Začátek	Adresa	Funkce	Data	LRC	Konec
znak “:“	2 znaky	2 znaky	0 až 2*252 znaků	2 znaky	2 znaky CR, LF

Obr. 4: Zpráva v režimu ASCII

Vlastnosti přenosu (10 bitů):

1 start bit

7 datových bitů

1 bit parita

1 stop bit

MODBUS na sériové lince

MODBUS Serial Line protokol je komunikace typu Master-Slave. V jeden okamžik na sběrnici může být jeden master a 1-247 slave jednotek. Komunikaci vždy začíná master, slave nesmí bez pověření vysílat. Master nemá adresu, za to slave jednotky musí mít jedinečnou v celé síti. Na fyzické úrovni modelu můžou být použita různá sériová rozhraní, například RS-232.

Vysílání může být ve dvou režimech:
- unicast režim – master adresuje jedné slave jednotce
- broadcast režim – master adresuje všem jednotkám.

MODBUS RTU

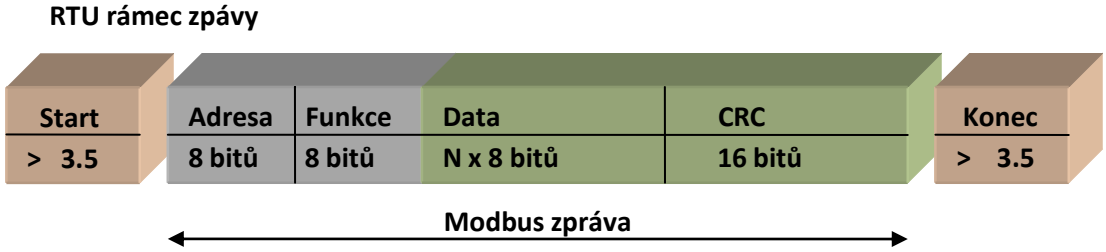
V režimu RTU je 8 - bitový bajt zprávy dva 4 - bitové hexadecimální znaky (Př.: hexadecimální hodnota “3E” je poslána jako dva čtyřbitové znaky “0000 0011“). Mezery mezi znaky nesmějí být delší než 1.5 znaku. Začátek a konec zprávy je definován pomlkou delší než 3.5 znaku.

Vlastnosti přenosu (11 bitů):

1 start bit
8 datových bitů
1 bit parita
1 stop bit

Každá jednotka musí podporovat sudou paritu. Pokud není použita parita, je nahrazena druhým stop bitem.

2.5 Modbus zpráva

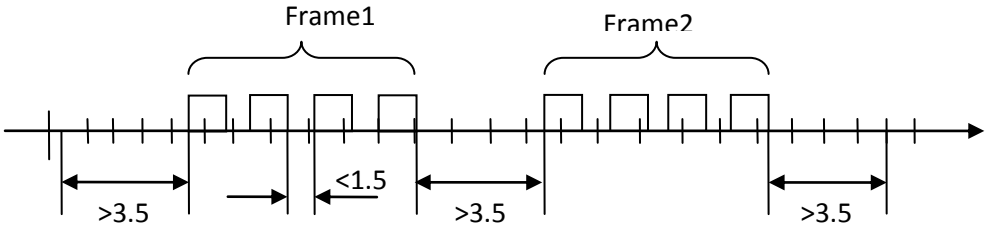


Obr. 5: RTU rámec zprávy

Na obrázku 5 je rámec modbus zprávy, kde je vždy na prvním místě adresa následována funkčním kódem (obě pole jsou velikosti 1 bajt). Po té následují data, která můžou mít proměnnou délku. Posledním polem v rámci je dvou-bajtovy kontrolní součet CRC (viz 2.6.2). Maximální velikost Modbus RTU rámce je 256 bajtů, pro data tedy zbývá maximálně 252 bajtů.

Časování přenosu

Pokud se vyskytne mezera mezi znaky kratší než 3,5 násobku znaku a delší než 1,5 násobku znaku, je rámec přijímacím zařízením považován za neplatný a čeká na další platný rámec (Obr. 6).



Obr.6: Ukázka časování přenosu znaků a rámců

Modbus zpráva je vysílajícím zařízením uložena do rámce, u kterého víme, kde začíná i končí, protože jsou tyto zprávy rámce odděleny intervalem přinejmenším 3.5 znaku času(v RTU módu). Toto implementujeme jako znak času vzhledem k přenosové rychlosti, která je použitá v síti. To dovoluje přijímajícímu zařízení, aby začalo na začátku zprávy a vědělo, kdy zpráva končí. Pokud není zpráva kompletní, musíme jí odchytit a podle toho vyvolat výjimku. Přenášené znaky jsou hexadecimální: 0–9, A–F. Po posledním přeneseném znaku, následuje opět interval přinejmenším 3.5 znaku času značící konec zprávy. Nová zpráva může začít po tomto uplynutí tohoto intervalu. Celý rámec musí být vysílán jako postupný tok znaků, přičemž mezera mezi jednotlivými znaky nesmí přesáhnout více než 1,5 násobku znaku. Je-li mezera mezi znaky delší než 1,5 násobku znaku a kratší než 3,5 násobku znaku, je nastavena chybová hodnota ve finálním CRC poli, neplatnou kombinací zpráv.

2.6 Používané kontroly chyb

Při přenosu dat v módu Modbus RTU se pro zabezpečení přenosu používají tyto kontroly chyb:

2.6.1 Parita

Paritní bit – slouží pro zpětnou kontrolu přijatých dat. Každé zařízení s protokolem Modbus musí povinně umožňovat kontrolu sudé parity. Další variantu (lichá parita, žádná parita) lze použít, avšak už nejsou povinné. Jedná se o doplnění datových bitu o paritní bit tak, aby součet všech logických 1 (jak data tak parita) byl sudý nebo lichý - takto lze detekovat chybu v jednom bitu, kdy při přenosu po lince dojde k chybě. Při použití metody kontroly bez parity je paritní bit nahrazován stop bitem, tedy zpráva obsahuje 8 datových bitů a 2 stop bity.

2.6.2 CRC

Kontrolní součet je další metodou pro zjištění chyb v přenášených datech. K detekci chyb slouží 16-bitové CRC pole s generujícím polynomem x16 + x15 + x2 + 1. CRC – je kontrolní slovo, které obsahuje kód vytvořený ze všech bajtů v rámci, počínaje adresou zařízení. Jeho velikost v Modbus zprávě jsou 2 znaky, které obsahují 16-bitovou binární hodnotu. Nižší bajt kontrolního slova je posílán jako první a po něm následuje vyšší bajt po kterém musí následovat 3,5 znaku času mezera. CRC je spočítáno vysílajícím zařízením, které ho přidá na konec vysílaného rámce. Přijímací zařízení spočítá CRC z přijatých bajtů a porovná ho s přijatým CRC ve zprávě. Jsou-li oba kódy totožné, zpracuje přijatý rámec (pokud je adresovaný jemu) dle funkčního kódu. Nejsou-li kódy totožné, přijatý rámec není akceptován a přijímací zařízení čeká na další rámec.

2.7 Záporná odpověď

Server může vyvolat na základě požadavku tyto situace:

- Pokud server přijme bezchybně požadavek a je normálně zpracován, vrátí klientovy normální odpověď.
- Když požadavek nepřijme kvůli komunikační chybě, není vrácena žádná odpověď. U klienta dojde k vypršením časového limitu pro příjem odpovědi.
- Server přijme požadavek, ale detekuje komunikační chybu (CRC,..), nevrací žádnou odpověď. U klienta dojde k vypršením časového limitu pro příjem odpovědi.
- Jestliže server přijme bezchybně požadavek, ale nemůže ho normálně zpracovat, vrátí klientovi zápornou odpověď (viz Tab. 2)

kód	funkce	data
01	ilegální funkce	funkce není serverem podporována
02	ilegální adresa dat	zadaná adresa je mimo rozsah
03	ilegální hodnota dat	požadovaná data jsou neplatná
04	selhání zařízení	při požadavku došlo k neodstranitelné chybě
05	potvrzení	požadavek přijat, vyřízení bude trvat déle
06	zařízení je zaneprázdněné	server je zaneprázdněn dlouhým požadavkem
08	chyba parity	chyba parity při čtení souboru
0A	přenosová cesta nedostupná	přetížení interní cesty
0B	cílové zařízení neodpovídá	cílové zařízení neodpovídá

Tabulka č.2: Tabulka chybových kódů

3 Přípravek PPZ082

3.1 Popis přípravku

Přípravek PPZ 082 je určen pro nastavení a přeměření analogových převodníků tlaku KP 053 a KP070. Skládá se z desky elektroniky se svorkami pro elektrické připojení převodníků a rozvaděče tlaku se šroubeními pro upevnění převodníků a připojení na tlak. V přípravku jsou multiplexery pro výběr měřeného převodníku, stabilizátor napájecího napětí s omezením proudu, měřicí procesor s AD převodníkem a obvod FT232R firmy FTDI pro připojení na rozhraní USB. Obvod FT232R je nakonfigurován pomocí programu MProg (http://www.ftdichip.com/Resources/Utilities.htm#MProg). Každý přípravek má přiřazeno individuální výrobní číslo (Fixed Serial Number) tvaru PPZ82-xx, kde xx je přadové číslo přípravku. Obvod se nemapuje v systému jako virtuální sériový port, ale je třeba použít přímý přístup přes knihovnu D2XX (viz http://www.ftdichip.com/Drivers/D2XX.htm). Měřený převodník je vybírán nastavením adresy jeho pozice (0 až 15) pomocí výstupů CBUS0 až CBUS3 obvodu FT232R pomocí funkce knihovny FT_SetBitMode. Signál sériového portu DTR ovládaný pomocí funkcí knihovny FT_SetDtr a FT_ClrDtr slouží k přepínání mezi měřením analogových převodníků (DTR=0) a digitálních převodníků (DTR=1). Přepínání zajišťuje multiplexer. Při měření analogových převodníků je komunikace připojena k měřicímu mikroprocesoru. Měřicí mikroprocesor změří hodnotu napětí Up, Ut a Vcc vybraného převodníku a vrátí naměřené hodnoty. Měřicí procesor komunikuje protokolem Modbus, nastavení komunikace: 38400 bd, 8 data bitů bez parity.

Po přečtení dokumentace k přípravku PPZ082_04[9] a Modbusu[3] jsou rozdělené do dvou kategorií podle protokolu Modbus:

3.2 Uživatelsky definované funkce

U těchto funkcí je v poli adresy každého rámce nastavena hodnota 248 a v poli specifikující funkci hodnota 110.

- **Čtení okamžitých hodnot**

Po dotazu mikroprocesor spustí měření a čeká na jeho dokončení. Po dokončení měření vysílá odpověď. Prodleva mezi dotazem a odpovědí závisí na rychlosti měření. Detail zprávy je na obr. 7.

Dotaz	248	110	R				
Odpověď	248	110	R	12	Up	Ut	Vcc

Obr.č.7: Formát zprávy čtení okamžitých hodnot

R rychlost měření jako **char**: 7 .. 20ms, 8 .. 250ms
Up měřená hodnota napětí Up jako **float** (4 bajty LSB první, MSB poslední)
Ut měřená hodnota napětí Ut jako **float** (4 bajty LSB první, MSB poslední)
Vcc měřená hodnota napětí Vcc jako **float** (4 bajty LSB první, MSB poslední)

- **Čtení EEPROM**

Po dotazu mikroprocesor přečte jednu stránku (32 bajtů) z připojené paměti EEPROM (Obr. 8). Čtení trvá 30ms. Po dokončení odešle v odpovědi data. V případě, že paměť nelze číst (není připojena apod.) vrací exception response.

Dotaz	248	110	10	Adr				
Odpověď	248	110	10	32	B0	B1	...	B31

Obr.č.8: Formát zprávy čtení EEPROM

Adr adresa prvního čteného bajtu v EEPROM (0 až 511dec) jako **int** (2 bajty LSB první)
B0 až B31 jednotlivé bajty vyčtené z EEPROM

- **Zápis EEPROM**

Po dotazu mikroprocesor zapíše jednu stránku (32 bajtů) do připojené paměti EEPROM a zapsaná data zkontroluje (Obr. 9). Zápis a kontrola trvají 50ms.
Po dokončení odešle odpověď. V případě, že paměť nelze zapsat nebo byla zjištěna chyba při kontrole vrací exception response s kódem 4.

Dotaz	248	110	9	Adr	32	B0	B1	...	B31
Odpověď	248	110	9						

Obr.č.9: Formát zprávy zápisu EEPROM

Adr adresa prvního čteného bajtu v EEPROM (0 až 511dec) jako **int** (2 bajty LSB první)
B0 až B31 jednotlivé bajty pro zápis do EEPROM

3.3 Veřejné požadované funkce

Veřejné kódy požadovaných funkcí:

Zadavatel po mě požaduje minimálně tyto vybrané veřejné funkce:
0x03 Read Holding Registers
0x05 Write Single Coil
0x10 Write Multiple Registers
0x11 Report Slave Id

Popis těchto veřejných funkcí:

- **03 (0x03)** Read Holding Registers

Slouží ke čtení obsahu souvislého bloku až 125 uchovávacích registrů. V požadavku je zapotřebí specifikovat adresu prvního registru a počet registrů. A v požadované odpovědi odpovídá každému registru dvojice bajtů (Obr.10).

Požadavek

Kód funkce	1 bajt	0x03
Počáteční adresa	2bajt	0x0000-0xFFFF
Počet registrů	2 bajty	1-125(0x7D)

Odpověď

Kód funkce	1bajt	0x03
Počet bajtů	1bajt	2*N
Hodnoty registrů	2*N(počet registrů) bajtů	

Chyba

Kód funkce	1bajt	0x83
Chybový kód	1bajt	01,02,03,04

Obr.č.10: Formát zprávy čtení uchovávacích registrů

- **05 (0x05)** Write Single Coil

Slouží k nastavení jednoho výstupu na vzdáleném zařízení do stavu ON nebo OFF. V požadavku specifikujeme adresu výstupu nastavením hodnot 0x0000…. OFF, 0xFF00 ….. ON. Normální odpověď je kopií požadavku.(Obr. 11)

Požadavek

Kód funkce	1 bajt	0x05
Adresa výstupu	2bajt	0x0000-0xFFFF
Hodnota výstupu	2 bajt	0x0000 nebo 0xFF00

Odpověď

Kód funkce	1bajt	0x05
Adresa výstupu	2bajty	0x0000-0xFFFF
Hodnota výstupu	2bajty	0x0000 nebo 0xFF00

Chyba

Kód funkce	1bajt	0x83
Chybový kód	1bajt	01,02,03,04

Obr.č.11: Formát zprávy zápisu jedné cívky

- **16 (0x10)** Write Multiple Registers

Slouží nám k zápisu bloku až 120 registrů. V požadavku je specifikace adresy prvního registru, který se má zapisovat, počet registrů a hodnoty, které se mají zapsat. Normální odpověď obsahuje počáteční adresu a počet zapsaných registrů (Obr. 12).

Požadavek

Kód funkce	1 bajt	0x10
Počáteční adresa	2bajt	0x0000-0xFFFF
Počet registrů	2 bajty	1-120(0x78)
Počet bajtů	1bajt	2*N
Hodnoty registrů	2*N(počet registrů)bajtů	

Odpověď

Kód funkce	1bajt	0x10
Počáteční adresa	2bajty	0x0000-0xFFFF
Počet registrů	2bajty	1-120(0x78)

Chyba

Kód funkce	1bajt	0x90
Chybový kód	1bajt	01,02,03,04

Obr.č.12: Formát zprávy zápisu více registrů

- **17 (0x11)** Report Slave Id

Slouží k zjištění typu zařízení, stavu a dalších informací o zařízení. Konkrétní obsah odpovědi závisí na typu zařízení.

Požadavek

Kód funkce	1 bajt	0x11

Odpověď

Kód funkce	1bajt	0x11
Počet bajtů	1bajt	
ID zařízení	Závislé na zařízení	
Identifikátor běhu	1bajt	0x00=off,0xFF=on
Další data		

Chyba

Kód funkce	1bajt	0x91
Chybový kód	1bajt	01,04

Obr.č.13: Formát zprávy sdělení identifikace

4 FTDI

Pokud bychom pracovali s klasickým sériovým rozhraním, použili bychom třídu SerialPort z knihovny System.IO nacházející se v knihovně tříd rozhraní .NET Framework. Představuje prostředek sériového portu pro vytvoření spojení a práci s ním. Vzhledem k podobným vlastnostem, je dobré se seznámit nejprve s touto třídou.

4.1 SerialPort

Základní vlastností, je jméno sériového portu, na který se budeme připojovat: PortName, a rychlost přenosu dat – Baud Rate. Paritní bit – slouží pro zpětnou kontrolu přijatých dat. Nejčastější parita je sudá a lichá, jedná se o doplnění datových bitu o paritní bit tak, aby součet všech log.1 (jak data tak parita) byl sudý nebo lichý - takto jde detekovat chybu v jednom bitu, kdy při přenosu po lince dojde k chybě
př. - Data = 00111011, sudá parita je P = 1 a lichá P =0. StopBits-kterým přenášený rámec dat končí. DataBits-nastavuje standardní délku datových bitů na bajt (8). Toto nastavení musí být stejné i ve druhém zařízení, ke kterému bude počítač pomocí sériové linky připojen.

4.2 FTDI

V přípravku PPZO82 je využit obvod FT232RL firmy FTDI(Firma FTDI byla založena roku 1990 a sídlí ve skotském Glasgow a specializuje se na konverzi klasických periferií PC na USB). Sériový port RS232 se pomalu stává v řadě případů nedostupný a je třeba používat novější rozhraní. Protože jsou ale jednočipové mikropočítače velmi dobře přizpůsobeny pro sériovou komunikaci, tak se dobrým řešením stává právě převodník USB na sériový kanál. Ovladače, programy a originální dokumentace jsou ke stažení na stránkách výrobce čipů FTDI[6]. Tyto obvody jsou podporované ve všech rozšířených operačních systémech.

Typicke aplikace využívající tyto převodníky[5][str.2]:

USB to RS232 / RS422 / RS485 Converters USB Industrial Control
USB MP3 Player Interface
USB FLASH Card Reader / Writers
Set Top Box PC - USB interface
USB Digital Camera Interface
USB Wireless Modems

4.3 FT232R

4.3.1 Specifikace

FT232R je poměrně nový obvod od firmy FTDI[6], který má některé výhody oproti předchozím verzím: Potřebuje méně externích součástek, konfigurační paměť EEPROM je integrovaná uvnitř obvodu, jsou dostupné synchronní a asynchronní bitbang režimy. Zajímavá je i cena, podpora řízení přenosu HW i SW, podpora úsporných režimů a probouzení PC. Obvod je dopředu nejprve nakonfigurován pomocí dostupného softwaru. Lze u něj nastavit standardní parametry jako např. 7 nebo 8 bitů, stopbit, parita, baudrate(300Bd až 3MBd). Velikosti bufferu jsou 256B na přijímací straně a 128 B na vysílací straně. Konfigurace se zapisuje do interní paměti EEPROM. FT232R přidává dvě nové funkce v porovnání se svými předchůdci. První je nakonfigurování linek CBUS jako zdroje výstupního obdélníkového hodinového signálu s kmitočty-6MHz, 12MHz, 24MHz, a 48Mhz, které mohou být vyvedeny mimo zařízení a slouží k řízení mikrořadiče nebo externí logiky. Druhou je jedinečné číslo (FTDIChip-ID™), které je nahráno do zařízení během výroby a je čitelné přes USB, což tvoří základ bezpečnostního klíče, který lze použít k ochraně aplikačních programů proti kopírování.

4.3.2 Integrovaná EEPROM

Tato integrovaná EEPROM slouží k uložení nejrůznějších parametrů obvodu, mezi něž patří např. také nakonfigurování linek CBUS. Předchozí generace FTDI USB pokud používala USB VID, ID ,PID, sériové číslo a informace o produktu jiných než výchozích hodnot, tak zařízení požadovala externí EEPROM. Tato externí EEPROM je nyní integrovaná přímo do čipu FT232R, proto všechny zařízení mohou měnit tyto informace o produktu. Vnitřní EEPROM je také k dispozici pro uložení dalších dat a je programovatelná v obvodu přes USB. Naprogramovaná EEPROM - FT232R je dodáván s interní EEPROM s přednastaveným sériovým číslem, které je jedinečné pro každé zařízení. To ve většině případů odstraní potřebu programovat EEPROM přístroje.

4.3.3 Integrovaný obvod hodinového signálu

Předchozí generace USB FTDI UART zařízení vyžaduje externí krystal nebo keramický rezonátor. Obvod pro generování hodinového signálu je nyní integrován do zařízení.

4.3.4 Bit Bang

Výrobce obvodů FTDI Chip implementuje do všech svých řadičů tzv. Bit Bang mód. Tento režim umožňuje pomocí PC přímý zápis na signálové vývody,takže se chovají jako GPIO u různých mikroprocesorů či mikrokontrolérů. Přepnout do tohoto režimu můžeme obvod buď programově nebo pomocí manuální konfigurace EEPROM paměti USB řadiče.

Synchronní Bit Bang Mode - synchronní „bit bang“ režim se od asynchronního režimu liší tím, že pokud zařízení zapisuje tak jsou piny rozhraní jen pro čtení. To usnadňuje ovládáním programu změřit odezvu synchronních dat vrácených na výstupu.Tato funkce byla již dříve použita u FT2232C čipech.

Asynchronní Bit Bang Mode - V tomto režimu můžeme datové linky libovolně konfigurovat jako vstupy nebo výstupy. Rychlost zápisu/čtení vychází z nastavení Baud Rate generátoru. Po zápisu dat se na výstupních linkách objeví zapisované slovo, které je na nich drženo až do té doby, kdy je přepsáno jinými daty.

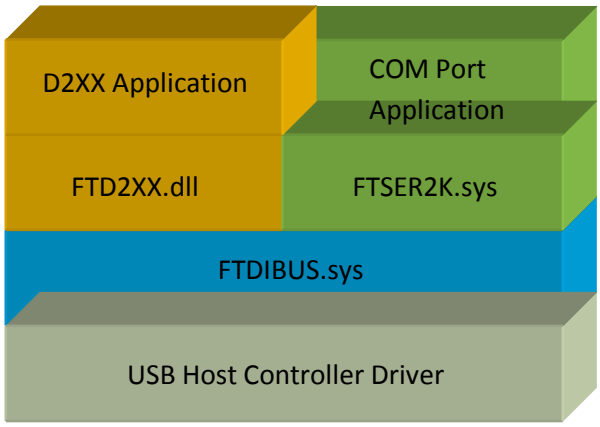
CBUS Bit Bang Mode - Tento režim je podobný asynchronnímu bitovému režimu. Dovoluje čtyři CBUS piny nastavit jako GPIO piny. Můžeme tento režim použít zatímco je používáno UART rozhraní, a tím zajistit až čtyři univerzální I/O piny, které jsou k dispozici v průběhu normálního provozu.

4.3.5 FTDICHIP-ID™

Každému FT232R čipu je přiděleno jedinečné číslo, které je nahráno do zařízení ve výrobě. Po tom již nelze přeprogramovat výrobcem ani uživatelem. To výrobci umožňuje prodávat FT232R pod licencemi softwaru na základě klíče. FTDIChip-ID™ čísla jsou šifrovány s ostatními informacemi a je uloženo v uživatelské části vnitřní EEPROM. Kontrolu je možno provést dešifrováním tohoto klíče, čímž získáme jeho hodnotu.

4.4 Ovladače

Distribuované ovladače lze rozdělit na následující: VCP, D2XX a CDM Combined Driver Model. Obrázek 14 nám přibližuje stav, kdy jsou k dispozici obě rozhraní (výchozí konfigurace). Nelze *současně* využívat obě výše uvedená rozhraní. Ovladače jsou všechny k dispozici ke stáhnutí zdarma z webu FTDI. Samozřejmě jsou zde i ovladače pro jiné operační systémy viz stránky FTDI[6].



Obr.č.14: Windows CDM Driver Architecture

4.4.1 VCP

Rozhraní Virtual COM Port umožňuje aplikacím komunikovat s převodníkem jako se standardním sériovým portem. Pro komunikaci je možno použít libovolný terminálový program a vybrat si nově vzniklý COM port. Na rozdíl od obyčejného COM portu dojde k přerušení spojení mezi programem (terminálem) a USB COM portem kdykoli odpojíme a připojíme USB zařízení.

4.4.2 D2XX Direct Driver

Rozhraní D2XX API je proprietární rozhraní firmy FTDI a slouží pro komunikaci s obvody FTDI včetně ovládání jejich speciálních funkcí (např. zápis dat do paměti EEPROM, nastavení zařízení do jiného módu). Ke komunikaci s FTDI zařízením stačí přidat do projektu knihovnu FTD2XX_NET, která je volně šiřitelná a zajišťuje přístup k funkcím či proměnných, potřebných pro práci s tímto rozhraním. Podrobnosti nalezneme v dokumentu [1], který poskytuje vývojářům potřebné informace pro práci s FTD2XX knihovnou.

4.4.3 CDM Combined Driver Model

Pouze pro Windows, instalace obou typů driverů (D2XX, VCP) najednou, ale v jednom okamžiku lze pracovat pouze s jedním.

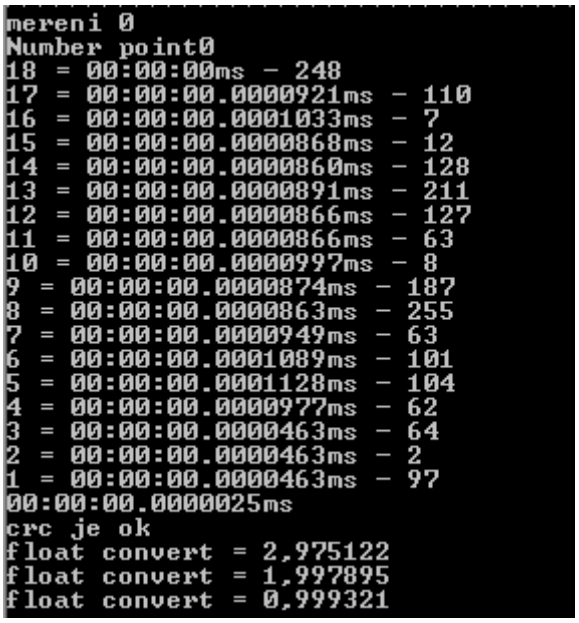
5 Implementace

Pro naprogramování aplikace byl zvolen objektový programovací jazyk C# v implementačním prostředí Visual Studia 2010. První verze aplikace byla navržena jako konzolová aplikace - CLR Console Application součást platformy Microsoft .NET Framework. Konzolová aplikace má pouze textový výstup, bez grafického uživatelského rozhraní a běží v příkazovém řádku, tudíž nedokáže zobrazovat grafiku. Je proto vhodná pro účely ladění.

V konzolové aplikaci byla vytvořena testovací třída Modbus. V této třídě byla nejprve naimplementována metoda navazující připojení Connect (5.5.1), připojující zařízení podle jejího sériového čísla. Následovala funkce pro sestavení zprávy BuildMessage (5.3.1) a kontroly CRC (5.3.2). Po prozkoumání funkcí zadané zadavatelem byli do BuildMessage přidány parametry, které vkádají do datové části zprávy bajt podfunkce, nebo jiné funkce podle volané metody. Ze základních funkcí chyběla již jen metoda pro získání odpovědi a vyhodnocení. Tady nastal problém, jak odchytávat mezi znakové mezery. Výborně k tomu posloužila na konec třída StopWatch z knihovny System.Diagnostic, která dokáže měřit délku času v milisekundách. V metodě vyhodnocení je potom ve zprávě isolována datová část a s použitím vhodného algoritmu rozdělena na související části, které převádí díky knihovnám BitConverter (System.BitConverter) a Parse (Int32.Parse) na desetinný tvar. V další etapě byly implementovány funkce zadané zadavatelem. Tyto funkce propojují všechny základní metody popsány výše, jejichž výstupem je dotaz do připojeného zařízení.

Jestliže tedy dojde v programu k volání metod, pro zasílání požadavků do FTDI zařízení ze třídy Modbus, je nejprve volána metoda buildMessage pro sestavení zprávy. Návratovou hodnotou této funkce jsou bajty vlastní zprávy, včetně kontrolního součtu CRC, určené k odeslání. Tuto zprávu odesílá mateřská metoda ihned po sestavení jako dotaz do zařízení FTDI. Samotné odeslání zprávy je spuštěno ve vlastním vlákně, které je potom uspáno na délku 3.5 času znaku, kdy dojde k potřebné pomlce mezi jednotlivými zprávami. Pokud vše proběhne v pořádku je volána metoda response pro získání odpovědi ze zařízení. Zde se přijímají jednotlivé bajty z bufferu a přirom se kontrolují zda nebyli překročeny časy 1.5 a 3.5 znaku času, které značí chybnou zprávu nebo její konec. V této metodě také probíhá opět kontrola CRC, zda přišla zpráva celá. Jako návratová hodnota je pole bajtů, obsahující odpověď na zaslaný dotaz. Pokud je vše v pořádku je zavolána metoda toValues (5.5.3) pro vyhodnocení přijeté zprávy a která patřičné bajty převede na desetinné

čísla. Ukázka výstupu volání metody ReadInstantValues s naměřenými časy je na obrázku 15.



Obr.č.15: Konzolová aplikace

Po vyladění modulu Modbus byla aplikace doplněna o graficé uživatelské rozhraní. Prezenční uživatelská část využívá WindowsForms - aplikační programové rozhraní pro formulářová aplikace s bohatým grafickým uživatelským rozhraním (Graphics User Interface, GUI), které je součástí .NET Framework společnosti Microsoft, poskytující přístup k prvkům rozhraní systému Microsoft Windows. Hlavním prvkem aplikace určené pro Microsoft Windows pro prezentaci dat je formulář. Na tento formulář jsou umístěny základní stavební bloky uživatelského rozhraní (panel, GroupBox), které fungují jako kontejnery pro umístění dalších ovládacích prvků.

5.1 Způsob implementace

Základem návrhu je třída Modbus, která se dělí na částečné (partial) třídy ModbusGeneral tvořící základ pro práci s Modbus rámci a ModbusFunction obsahující zpřístupňující metody pro zadané funkce. Tyto částečné třídy nám umožňují definovat třídu do více souborů a tím rozdělit kód na části. Kompilátor potom spojí všechny patřičné soubory do jedné třídy, které potom vystupují jako třída jedna. Tyto třídy jsou přidány jako reference k hlavnímu formuláři aplikace, který obsluhuje události vyvolané uživatelem a ostatní důležitou logiku zajišťující celkovou funkčnost aplikace.

Hlavní metody tříd:

Třídu ModbusGeneral

Tu tvoří metody (jsou blíže přiblíženy v části 5.3):

ConectSerial – metoda, která navazuje spojení s FTDI zařízením, nejprve načte seznam připojených zařízení a po té vytvoří připojení s vybraným FTDI zařízením ze seznamu

CloseSerial - metoda ukončující spojení se zařízením

BuildMessage - tato metoda sestavuje zprávu odesílanou do FTDI zařízení jako požadavek

Response - v této metodě získáme jako návratovou hodnotu odpověď z FTDI zařízení na předcházející dotaz

CRC - metoda, která vrací kontrolní součet poslané/příjmuté zprávy

Control – ta slouží ke kontrole přijatých zpráv, a v případě odchycení chyby tuto chybu identifikovat.

Třída ModbusFunction

Tyto funkce metod jsou popsány v zadání od zadavatele (3.3) a jejich implementace v části 5.4

ReadInstantValues - funkce čtení okamžitých hodnot, které získá po dotazu na měření mikroprocesoru v připojeném zařízení

ReadEeprom - tato metoda čte jednu stránku (32bajtů) z připojené paměti EEPROM po dotazu do mikroprocesoru

WriteEeprom - po jejím zavolání mikroprocesor zapíše jednu stránku do připojené paměti EEPROM

ReadHoldingRegisters - ta slouží ke čtení obsahu vnitřních registrů

WriteSingleCoil - nastavuje jeden výstup spínaného zařízení do stavu ON nebo OFF

WriteMultipleRegisters - tato metoda slouží k zápisu bloku až 120 registrů

Report SlaveId - Slouží k zjištění typu, stavu a dalších informací o zařízení. Konkrétní obsah odpovědi závisí na typu připojeného zařízení

Prezenční část tvoří formulář **ApplicationForm,** ve kterém jsou naimplementovány tyto hlavní metody (podrobněji viz 5.5):

DefaultStartup - ta slouží k záklanímu nastavení parametrů FTDI připojení

Connect - metoda obsluhující připojení FTDI zařízení

Test - v této funkci probíhá vlastní měření pozic přípravku PPZ082

ToValues - zajišťuje získání naměřených hodnot z přijaté zprávy
Evaluated - pomocí této funkce probíhá vyhodnocení všech naměřených hodnot
WriteTxt - metoda ukládající výsledek měření do souboru

5.2 FTD2XX_NET

Jak už jsme se dozvěděli dříve ze zadání, je obvod FT232R nakonfigurován tak, že se nemapuje v systému jako virtuální sériový port, ale je třeba použít přímý přístup přes knihovnu D2XX [7]. Proto po prostudování dokumentace k FTDI[8] byla nejprve do projektu přidána reference na knihovnu FTD2XX_NET.dll, která zpřístupňuje metody, proměnné a konstanty potřebné pro práci s FTDI zřízením. Tyto metody jsou součástí logiky třídy Modbus, a pomáhají zajišťovat základní funkce celé aplikace. Na konec je přidán jmenný prostor FTD2XX_NET do všech tříd, kde je komunikace s FTDI zařízením vyžadována.

Použití knihovny FTD2XX_NET:

5.2.1 Stav zařízení

Při práci s FTDI zařízením používá knihovna D2XX pro kontrolu požadavků, nastavení parametrů, validaci metod aj. proměnnou FT_Status. Návratová hodnota této proměnné indikuje (ne)úspěch zadané operace. Při inicializaci ja nastavena na hodnotu “FT_OK“. Pokud vše proběhne v pořádku, její hodnota se nemění a naopak dojde-li k výskytu chyby, vrací pomocí konstatnt identifikující problém chybový kód (FT_DEVICE_NOT_OPENED, FT_IO_ERROR, FT_INVALID_PARAMETER, FT_INVALID_BAUD_RATE, FT_OTHER_ERROR aj.). V projektu je použita jak při kontrole volaných metod (read, write) tak i nastavení parametrů připojení (baudrate,timeout..). Způsob kontroly je patrný ze zdrojového kódu 1.

```
try
{
    ftStatus = myFtdiDevice.Write(WEEmessage, WEEmessage.Length, ref ftdiDeviceCount);


    if (ftStatus != FTDI.FT_STATUS.FT_OK)
    {

        Console.WriteLine("Nelze zapsat do zařízení( " + ftStatus.ToString() + ")");
        Console.ReadKey();
        return;
    }
    response();
}
catch (Exception e)
{
    error = "Nezdařilo se zapsat" + e.Message;
}
```

Zdrojový kód 1: Průběh kontroly nastavení FTDI

5.2.2 Připojení FTDI zařízení

Zařízení FTDI, může být specifikováno sériovým číslem, popisem nebo umístěním. Knihovna D2XX nabízí specifické metody pro spojení s tímto zařízením, které zvolíme podle použité identifikace používaných zařízení. Protože každý z přípravků PPZ082 má přiřazené individuální výrobní číslo (viz 3.1) “Fixed Serial Number“ tvaru PPZ82-xx, kde xx je pořadové číslo přípravku, vybral jsem funkci FT_ OpenBySerialNumber (serialNumber). Pomocí této funkce otevřeme zařízení podle sériového čísla přiřazeného každému z přípravků. Toto sériové číslo získáme pomocí metody FT_ListDevices(). Tato metoda vrací seznam připojených FTDI zařízízení , z kterého lze vyčíst počet připojených, sériová čísla a popisy těchto zařízení.

5.2.3 Zápis a čtení dat z FTDI zařízení

Pokud je zařízení otevřeno můžeme použít metodu pro zápis dat FT_Write(Buffer,Bytes,BytesWr) a pro čtení FT_Read(Buffer,Bytes,BytesR). Jako vstupní proměnná zapisující funkce je atribut Buffer ukazatelem na data určená k zápisu do FTDI zařízení, atribut Bytes – označuje počet zapisovaných bajtů a BytesWr - je ukazatel na proměnou, do které bude uložen počet zapsaných bajtů. Atribut BytesWr lze také použít pro kontrolu, zda byly zapsány všechny zaslaná data. Funkce pro čtení FT_Read čte data z FTDI zařízení, kde první atribut je opět ukazatel na buffer, do kterého se uloží přijatá data, následuje počet bajtů které chceme přečíst a poslední je ukazatel na proměnou dokteré bude uložen počet přijatých bajtů ze zařízení. Pokud se vyskytne při čtení/zápisu konflikt je do proměnné FT_Status uložena hodnota FT_IO_ERROR.

5.2.4 Ostatní použité metody a vlastnosti knihovny FTD2XX_NET

FT_Device_Info_Node - pomocí této metody získáme všechna data o připojených zařízeních jako je typ zařízení, jeho id, popis a sériové číslo
SetBaudRate – vlastnost pomocí níž nastavíme přenosovou rychlost komunikace
SetTimeout – slouží k nastavení časového limitu zápisu a čtení v milisekundách. Můžeme nastavit časový limit např. na čtení z FTDI zařízení. Tato metoda je po tom ukončena po uplynutí této doby, nebo když je přečten zadaný počet bajtů, podle toho co je splněno dřív.
SetDTR(bool) - funkce ovládající signál sériového portu DTR slouží k přepínání mezi měřením analogových převodníků (false) a digitálních převodníků (true).
SetBitMode(Mask, Mod) – pomocí této funkce je vybrán měřený převodník nastavením adresy jeho pozice (0 až 15) pomocí výstupů CBUS0 až CBUS3 obvodu FT232R. Masku tvoří 8 -bitová hodnota a její volbou vybíráme jednotlivé měřené pozice. Její vyšší čtyři bity nastaví vlastnost linek CBUS3-CBUS0(0=vstup,1=výstup) a spodní čtyři bity nastaví na příslušné lince logickou hodnotu 0 nebo 1 v případě, že dotyčná linka je nastavená jako výstupní. Mod slouží k nastavení režimu viz 4.3.4., který je podle zadání nastaven na režim CBUS Bit Bang Mode.
SetDataCharacteristics – tato metoda je volána před vytvořením připojení s FTDI zařízením a nastavuje základní vlastnosti pro připojení, jako jsou: dataBits, stopBist, Parity.
dataBits - zastupuje počet bitů na slovo, může být 7 nebo 8
stopBits - je také počet bitů, ale kterým slovo končí, může být 1 nebo 2
Parity - paritní bit pro zpětnou kontrolu dat (viz 2.6.1). Ten může být nastaven na hodnotu FT_PARITY_NONE, FT_PARITY_ODD, FT_PARITY_EVEN
Způsob nastavení vlastností je nastíněn ve zdrojovém kódu 2.

```
mod.ftStatus = mod.myFtdiDevice.SetBaudRate((uint)Modbus_General.BaudRate.brate38400);
mod.ftStatus = mod.myFtdiDevice.SetDTR(false);
```

Zdrojový kód 2: Nastavení parametrů FTDI

5.3 Metody třídy ModbusGeneral

5.3.1 BuildMessage

Metoda buildMessage je metodou, která setavuje zprávu ze zadaných parametrů, kterou pošleme jako dotaz do FTDI zařízení. Je univerzální a má tyto vstupní parametry:

Adresa

pole adresa ve zprávě rámce obsahuje osm bitů (RTU). Platné slave adresy zařízení jsou v desetinné řadě 0 – 247. Pokud je zde nula, vysílá master zprávu všem zařízením. Individuální adresy slave zařízení jsou přiřazeny v řadě 1 – 247. V tomto projektu je použita rezervovaná adresa 248 z rezervovaných kódů adres.

Funkce

pole funkce obsahuje taky osm bitů (RTU). Pokud je zpráva poslaná z master zařízení k slave, tak funkční politika říká slave zařízení, jaká akce má být vykonána. Pokud slave zařízení odpovídá, tak používá funkční kódové pole k tomu, aby signalizovalo buď normální odpověď, nebo chybu. Pro normální odpověď, slave vrací stejný funkční znak v odpovědi. Pro zápornou odpověď, slave vrací kód, který vznikne přičtením hodnoty 80h(10000000) ke kódu funkce. Toho využijeme při odchytávání takovýchto záporných odpovědí.

Start, count

Některé funkce (5.4) implementované ve třídě ModbusFunction, vyžadují v odesílané zprávě bajt specifikující podfunkci požadavku a bajt plnící jinou funkci. Proto je metoda BuildMessage navržena univerzálně. Jsou zde pole start a count, která jsou zařazeny ve zprávě na třetí a čtvrté místo hned po adrese a funkci. Pokud nejsou využity, mají hodnotu null a místo nich je dosazeno do zprávy pole dat.

Pole dat

každý osmibitový bajt ve zprávě obsahuje dva hexadecimální znaky (00-FF). Ty jsou vytvořeny z jednoho RTU znaku. Datové pole zpráv poslané z master zařízení do slave obsahuje dodatečné informace, které zařízení slave musí použít pro danou funkci. Pokud nedojde k chybě, slave pošle požadovaná data do master zařízení. Pole dat může být i nulové délky, pokud funkce nepotřebuje žádné jiné informace.

Struktura zprávy:

<t> <slave adresa> <funkce> <data> <CRC> <toff>

<t>	časová prodleva delší než 3,5 znaku
<slave adresa>	adresa z rozsahu <1 … 247> 8 bitů
<funkce>	číselné označení funkce 8 bitů
<data>	význam je dán popisem jednotlivých funkcí N * 8 bitů
<CRC>	kontrolní součet (16 bitů)

```
public Byte[] buildMessage(byte address, byte function, byte start, byte count, byte[] data)
{

    byte[] message = new byte[data.Length + 6];
    message[0] = address;
    message[1] = function;
    message[2] = start;
    message[3] = count;
    int m = 4;
    for (int i = 0; i < data.Length; i++)
    {
        message[m] = data[i];
        m++;
    }
    byte[] finalCrc = CRC(message);
    message[message.Length - 2] = finalCrc[0];
    message[message.Length - 1] = finalCrc[1];
    return message;
}
```

Zdrojový kód 3: Metoda BuildMessage

Ze zdrojového kódu je patrné jak se postupně skládá celá zpráva. Nejprve vytvoříme bajtové pole zprávy dané velikosti. První bajt obsadí adresa, následuje pole specifikující požadovanou funkci. Další bajty jsou určené pro zadanou podfunkci nebo jiné požadované parametry. Potom procházíme v cyklu posílaná data a plníme datové pole zprávy. Na konci je metodou CRC spočítán kontrolní součet ze všech bajtů prozatím uložených ve zprávě, a je přidán na konec zprávy, kde zabírá poslední dva bajty. Výsledkem této metody je pole bajtů - zpráva.

5.3.2 Metoda CRC

Jak bylo popsáno již výše, slouží tato metoda k detekci chyb, kde se kontroluje 16 -bitové CRC pole. Vysílací zařízení spočítá CRC a přidá ho na konec vysílaného rámce. Přijímací zařízení spočítá CRC z přijatých bajtů a porovná ho s přijatým CRC ve zprávě. Protože se CRC skládá z kódu vytvořeného ze všech bajtů v rámci, musí být zpracován každý bajt zprávy.

Popis algoritmu CRC (zdrojový kód 4):

Tento algoritmus byl naimplementován podle postupu uvedeného v [2][str.41, 42]. Postupujeme tak, že na začátku nastavíme všech 16 bitů pole CRC na logickou jedničku. V metodě procházíme bajt po bajtu zprávou. Jako první se zpracuje první bajt zprávy což je adresa. Provede se logická operace exclusive-OR daného bajtu a CRC, výsledek je uložen do proměnné CRC. Ta je potom posunuta o 1 bit do prava(>>1). Pokud byl první bit před posunem nastavený na 1 provádí se logická operace exclusive-OR proměnné CRC s konstantní hodnotou 0xA001 - polynom. Celá operace posuvu se pak opakuje ještě 7 krát. Potom se zpracovává další bajt zprávy. Výsledek předchozího je vstupem místo samých jedniček do druhého výpočtu. Po zpracování posledního bajtu je výsledný CRC uložen za poslední bajt zprávy tak, že nejprve je uložen nižší bajt CRC a potom vyšší, rozdělený po 8 bitech.

```
public Byte[] CRC(byte[] dats)
{
    byte[] crc = new byte[2];
    int cr = 0xFFFF;
    int lsb;
    for (int i = 0; i < (dats.Length - 2); i++)
    {
        cr = cr ^ dats[i];
        int p = 8;
        while (p > 0)
        {
            lsb = cr & 0x0001;
            cr = cr >> 1;
            if (lsb == 1)
            {
                cr = cr ^ 0xA001;
            }
            p--;
        }
    }
    byte Hi = (byte)(cr >> 8);
    byte Lo = (byte)(cr & 0x00FF);
    crc[0] = Lo; crc[1] = Hi;
    return crc;
}
```

Zdrojový kód 4: Metoda CRC

5.3.3 Response

Další důležitou metodou, která je v aplikaci často užívána, je metoda pro získání odpovědi na požadavek v mikroprocesoru zařízení. Nejen že přijímá z paměti procesoru odpověď na předešlou akci, ale zároveň provádí kontrolu pomocí mezi znakových časů a metody Control. Po spuštění čeká do té doby, než budou bajty odpovědi připraveny ke čtení. Po té čte bajt po bajtu, které ukládá do výsledné proměnné. Při čtení je spuštěn časovač implementovaný třídou StopWatch z knihovny SystemDiagnostic, která poskytuje velice přesné měření uplynulého času, než proběhne čtení následujícího bajtu. Logika odchycení těchto časů je vytvořena podle grafu viz obr. 6. Pokud je při zpracování zprávy čtení jednoho bajtu delší než konstanta roundTime1_5, proběhne kontrola, zda také uplynula doba 3.5 znaku času. Je-li delší, potom je to považováno za konec zprávy. Proběhne kontrola metodou Control, zda došla v pořádku a probíhá vyhodnocení zprávy. Pokud je ale doba čtení větší než 1.5 násobek času a přitom menší než 3.5 násobek tohoto času, je to považováno za chybnou zprávu. Projde-li rámec všemi kontrolami, můžeme výsledný rámec v navazující metodě ToValues vyhodnotit.

```
ftStatus = myFtdiDevice.GetRxBytesAvailable(ref numBytesAvailable);
ftStatus = myFtdiDevice.Read(mezi, 1, ref numBytesRead);
stopWatch.Stop();

if (stopWatch.ElapsedMilliseconds > roundTime1_5)
{

    if (stopWatch.ElapsedMilliseconds > roundTime3)
    {
        Console.WriteLine("3.5time: " + stopWatch.ElapsedMilliseconds.ToString()
            + "vetsi pauza3.5");
        Control(answer.ToArray());
        finalResponse = answer.ToArray();
        break;
    }
    response();
    Console.WriteLine("3.5time: " + stopWatch.ElapsedMilliseconds.ToString()
        + "vetsi pauza1.5");
    error = "3.5time: " + stopWatch.ElapsedMilliseconds.ToString() +
        "time mezi znaky > 3.5";
    break;

}
```

Zdrojový kód 6: Čtení v metodě Response

Ukázka časování přenosu znaků a rámců

Nejprve je třeba určit výpočet doby zpracování znaku času. Jeden bit je vyslán za (Time = 1 / BaudRate) sekundy. Znak je potom vyslán za (Time * 11(bitů)) sekundy, kde 11 je počet bitů v jednom znaku. Z tohoto času po té získáme násobky – 3.5 násobek času pro minimální mezeru mezi rámci a 1.5 násobek pro maximální mezeru mezi znaky. Například pokud je přenosová rychlost sériového přenosu 38400Bd, je jeden bit vyslán za 1/38400 s = 26 μs a pro vysílání/příjem jednoho znaku-bajtu to je 26 μs * 11 bitu = 286μs. Každému vyslanému rámci předchází pauza minimálně 3,5x delší než doba pro příjem jednoho znaku: 3,5 * 286 μs=1001 μs. To je identické s koncem rámce, kde musí být minimálně ta samá pauza. Celý rámec musí být vyslán jako postupný tok znaků a mezera mezi znaky nesmí být větší než 1,5 násobku znaku. Implementace výpočtu této časové mezery je zobrazena ve zdrojovém kódu 5. Vypočtená hodnota je vynásobena konstantou 1000 pro převod na milisekundy, ve kterých třída StopWatch měří potřebné časy. Vypočtená hodnata je na konec zaokrouhlena.

```
roundTime1_5 = Math.Round((1 / (double)(BaudRate.brate38400)) * 11 * (double)1.5 * 1000, 4);
```

Zdrojový kód 5: Výpočet časové mezery

5.4 Metody třídy ModbusFunction

ModbusFunction obsahuje metody vykonávající zadané funkce zadavatelem (5.4). V této aplikaci jsou použity pouze readInstantValues a readEEPROM. V rámci testování byla odzkoušena i metoda zápisEEPROM. Uvádím zde jen popis ReadInstantValues, protože implementace ostatních funkcí, které pro tuto aplikaci nejsou potřeba, je velmi podobná a není nutností je zde popisovat. Liší se pouze vstupními parametry a inicializací metody BuildMessage.

5.4.1 ReadInstantValues

Tato funkce slouží k získání okamžitých hodnot měřených pozic přípravku PPZ082. Vstupním parametrem této funkce je adresa, a zvolená rychlost měření, kterou nastavujeme prodlevu mezi dotazem a odpovědí. Tato rychlost je nastavena konstantou v této aplikaci na 250ms. Pokud byla nastavena na nižší hodnotu, bylo měření pozic zkreslené a je tedy použitelné pro jiná zařízení. Po předání parametrů metodě je nejprve sestavena zpráva požadavku. Zde je volána metoda pro sestavení zprávy BuildMessage s parametry adresa, požadovaná funkce a rychlost měření v datové části. Po sestavení zprávy, je vytvořeno nové vlákno, a v něm inicializována metoda FT_Write, zajišťující zápis dat do FTDI zařízení. Ihned po inicializaci je vlákno spuštěno.

Pomocí konstanty zastupující délku 3.5 násobku času, která je spočítána podle již známého vzorce, je inicializována metoda vlákna Join. Díky které toto vlákno počká a vytvoří tak žádanou mezi rámcovou mezeru mezi voláním metody pro získání odpovědi a slave zařízení takto identifikuje konec rámce. Pokud by zde tato mezera nebyla, bylo by možné poslat vice dotazů po sobě a tím znemožnit čtení slave zařízení. Na konci metody REadInstantValues je zavolána metoda response, která přečte data, zkontroluje zda je zpráva v pořádku a vrátí pole odpovědi zprávy .

```
RIVmessage = buildMessage(address, (byte)110, data, 0, nullable);
try
{
    Thread thr = new Thread(() => ftStatus = myFtdiDevice.Write(RIVmessage,
        RIVmessage.Length, ref ftdiDeviceCount));          //odeslani zpravy
    thr.Start();
    decimal time3 = Math.Round((1 / (decimal)(BaudRate.brate38400))
        * 11 * (decimal)3.5 * 1000, 3);
    int roundTime3 = (int)time3;
    thr.Join(roundTime3);
    if (ftStatus != FTDI.FT_STATUS.FT_OK)
    {
        Console.WriteLine("Nelze zapsat do zařízení( " + ftStatus.ToString() + ")");
        Console.ReadKey();
        return;
    }
    response();
}
```

Zdrojový kód 7: Část metody čtení okamžitých hodnot

5.5 Metody prezentační třídy ApplicationForm

V této třídě se nachází metody, propojující prezentaci dat uživateli s logickou částí aplikace. Najdeme tu funkce sprostředkující spojení s FTDI zařízením, nastavováním parametrů připojení, testování či vyhodnocení.

5.5.1 DefaultStartup, Connect

Funkce DefaultStartUp je umístěna do metody ApplicationForm_Load, která je spuštěna při prvotním načítání formuláře. Slouží k prvotnímu nastavení parametrů připojení jako je: čas zápisu/čtení, přenosová rychlost, pozice čtení. Po nastavení těchto parametrů, se pokouší aplikace připojit k FTDI zařízení. Implicitně je vybráno zařízení, které je první načtené pozici ze seznamu pomocí funkce FT_ListDevices(5.2). Je předpoklad, že ve většině případů bude připojen pouze jeden přípravek PPZ082 k pracovnímu PC. Pokud by bylo připojených zařízení více, je nutné vybrat požadované zařízení v menu aplikace, ve formuláři vlastnosti připojení.

Nedojde-li ke spojení s FTDI zařízením, je volána metoda Connect, navazující spojení v novém vlákně na pozadí, čekající na připojení zařízení. Connect nemá na starosti nic jiného, než připojení zařízení. Metoda běží ve smyčce, která končí navázáním spojením. Pokud se nepodaří vláknu navázat spojení s FTDI zařízením, je o tom uživatel informován a nelze spustit test přípravku do té doby, než zařízení připojí.

5.5.2 Test

Test je puštěn uživatelem tlačítkem “Start”, umístěném na hlavním formuláři. Pro spuštění tohoto tlačítka, je podmínkou připojené zařízení. Jestliže není zařízení připojené, není toto tlačítko aktivní. Po spuštění testu je nejprve vytvořena tabulka, do které budou uloženy naměřená data, po té zobrazovací prvek DataGridView, který bude tuto tabulku prezentovat uživateli. Jak už z názvu metody odpovídá, je vní zahájen test přípravku PPZ082. Spustí se funkce readInstantValues (2.4) ze třídy Modbus, která má za úkol čtení okamžitých hodnot pozic na přípravku. Pokud je vše v pořádku, proběhne čtení první pozice na přípravku a následně je spuštěna funkce ToValues (5.3.3) určená k vyhodnocení naměřených hodnot z vráceného pole odpovědi. Naměřené hodnoty jsou uloženy do tabulky a zobrazeny na formuláři pomocí DataGridView. Tím metoda končí a je volána znovu uživatelem, po stisknutím tlačítka pro pokračování ve čtení pozic, kde se v této metodě mění měřená pozice. Tímto způsobem proběhne postupně čtení všech šestnácti pozic na přípravku PPZ082.

5.5.3 ToValues

Tato metoda má jako vstupní parametr bajtové pole, získané z odpovědi metody pro čtení pozic přípravku ReadInstantValues (5.4.1). Nejprve odstraní z pole bajt adresy a funkce, vzápětí kontrolní součet CRC, tak že zůstane jen datová část zprávy. Tu potom zpracovává po čtyřech bajtech, kde každá skupina bajtů je jedna měřená hodnota. Výsledkem jsou hodnoty měřené pozice v desetiném tvaru, které jsou uloženy do tabulky výsledků a v nadřazené metodě přidány do DatagridView, který je zobrazí uživateli.

```
for (int u = vals.Length - 3; u > 3; u--)
{
    while (r > 0)
    {
        sb.Append(BitConverter.ToString(vals, u, 1));
        //Převede číselnou hodnota každého prvku zadaného subarray
        //bajtů na  řetězec v šestnáctkové soustavě.
        u--; r--;
    }
    string hexString = sb.ToString();
    UInt64 num = UInt64.Parse(hexString,
        System.Globalization.NumberStyles.AllowHexSpecifier);
    byte[] floatVals = BitConverter.GetBytes(num);//vrati pole bytu z cisla
    f = BitConverter.ToSingle(floatVals, 0);//prevede na desetine cislo
    row[s] = Math.Round(f, 3);//zaokrouhli na tri desetina cisla
    s--;
    sb.Clear();
    u++;
    r = 4;
}
table.Rows.Add(row);
```

Zdrojový kód 8: Část metody vyhodnocení hodnot

5.5.4 Evaluated

Evaluated jak už z názvu vypovídá vyhodnocuje naměřené hodnoty podle postupu popsaného v [9][str.3]. Přípravek PPZ je určen pro nastastavení a kontrolu převodníku tlaku. Nastavují se na něm hodnoty Up- napěťové zesílení a Ut- napětí na diodě těchto převodníků. Kontrola těchto přípravků tedy vyžaduje, zda měří tyto napětí správně. Mezní hodnoty pro jednotlivá napětí jsou pevně dány : Up(0.99-1.01), Ut(1.99-2.01), Vcc(2.90-3.10) a musejí být naměřitelná po připojení etalonu PPZ 08208 do přípravku na jednotlivých pozicích. Pokud je spuštěno vyhodnocení je nejprve vytvořen nový řádek DataGridView do kterého bude uložen výsledek. Potom algoritmus prochází celou tabulku s naměřenými hodnotami, kde kontroluje všechny tři naměřené hodnoty napětí. Pokud některá z těchto tří hodnot není v povoleném rozsahu, je ve formuláři nevyhovující řádek DataGridView obarven červeně. Jsou-li hodnoty v povolené mezi, řádku barva zůstává. Algoritmus takto prochází celou tabulku naměřených hodnot a na konci vypíše do nově vytvořeného řádku výsledek měření. Napěťové meze jsou zde nastaveny na pevno, neboť Sloupec, ve kterém je naměřená hodnota napětí mimo danou mez, je označen v řádku výsledku “CHYBA“, naopak sloupec se správnými hodnotami je označen ve výsledku “OK“.

5.5.5 WriteTxt

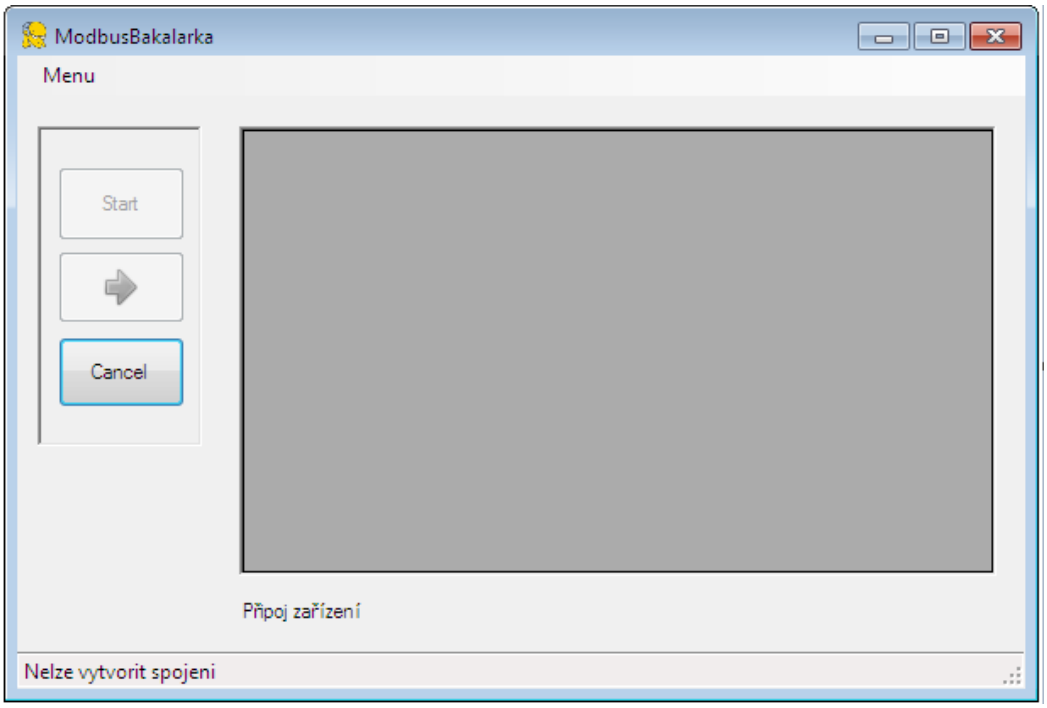
Jakmile je konec testu přípravku a probíhá metoda vyhodnocení evaluated, je po ní zavolána i metoda WriteTxt. Ta má za úkol, zapisovat výsledek testu do textového souboru “výsledek.txt“. Tento soubor je buďto vytvořen, pokud neexistuje, nebo jenom upraven tak, že je přidán záznam výsledku měření za poslední záznam z minulého měření. Ukládá se zde datum a výsledek vyhodnocení naměřených napětí spolu s identifikačním číslem přípravku (Obr.: 16), aby bylo možné se v tomto výsledku lépe orientovat. Pokud by bylo zapotřebí ukládat tyto informace do databáze, je možné tuto metodu rozšířit.

```
--- 19.2.2012 21:30:55   PPZ82-33  Up:OK  Ut:OK  Vcc:OK ---
--- 20.2.2012 20:36:11   PPZ82-33  Up:OK  Ut:OK  Vcc:OK ---
```

Obr.č.16: Výsledek

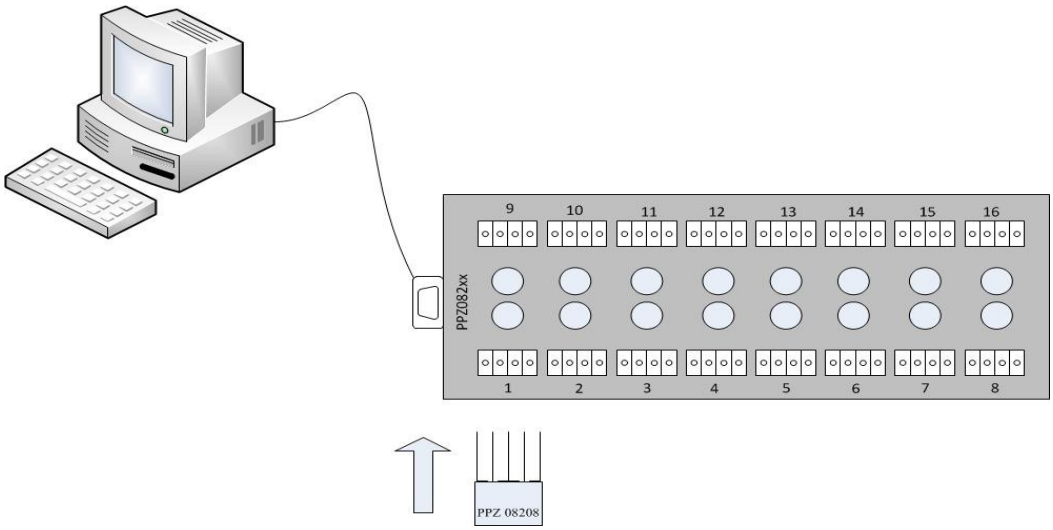
5.6 Aplikace

Jak už bylo naznačeno, aplikaci tvoří jeden formulář (Obr. 15). Po spuštění se zobrazí tento formulář, a uživatel má možnost stisknout jediné tlačítko start, čímž zahájí testování přípravku viz Příloha A. Celkem má k dispozici tři tlačítka, která jsou aktivní podle průběhu prováděného testu nebo potřeby.



Obr.č.17: Formulář aplikace

Je zde také možnost volby menu, kde je v novém formuláři na výběr nastavení hlavních vlastností FTDI připojení a informace o připojených zařízeních. Pokud je připojeno k pracovnímu PC více přípravků PPZ082, je zde možnost zvolit požadovaný přípravek z načteného seznamu. Vlastnosti, které lze nastavit jsou: přenosová rychlost, DTR - přepínání mezi měřením digitálních a analogových převodníků, parita a časový limit zápisu či čtení FTDI zařízení. Přípravek PPZ082 se může připojit před spuštěním aplikace tak i v průběhu jejího běhu, není-li již spuštěn test. Pokud je zařízení připojeno, je o tom uživatel informován v informační liště formuláře. Při jakémkoli přerušení spojení s FTDI zařízením, je vyvolána výjimka a oznámena uživateli zprávou. Aplikace se vrátí do počátečního stavu, a vyčkává na nové připojení. Pokud je spuštěn test, jsou změřeny hodnoty napětí na první pozici a uživateli je zpřístupněno tlačítko, pro přepnutí další pozice a její přeměření. Aplikace je tedy navržena poloautomaticky, a je vyžadována akce uživatele po každé změřené pozici přepnutím na další. Pokud by byl režim plné automatiky, uživatel by musel v určitém čase připojovat do svorek přípravku etalon PPZ 08208, což není někdy snadné. Proto je mezi měřením pozic ponechána časová nezávislost a vyžadována akce uživatele. Uživatel tedy v aplikaci postupně přepíná pozice a připojuje předepsaný etalon do svorkovnic přípravku (Obr. 16). Pokud je doměřena poslední šestnáctá pozice probíhá vyhodnocení naměřených hodnot. Uživateli je zobrazen výsledek testu a je uložen do textového souboru. Aplikace přejde do počátečního stavu, kde je očekávána výměna změřeného přípravku za jiný a spuštění nového testu.



Obr.č.18: Kontrola přípravku

Samotné nasazení aplikace je podle požadavků vytvořena spustitelná verze programu, a ta je umístěna na pracovním PC bez nutnosti instalace.

6 Testování aplikace

Aplikace byla testována již v části vývoje konzolové aplikace. V této části byla vylepšena univerzálnost metody BuildMessage pro sestavení zprávy, kde v datové části byli doplněny bajty doplňkových funkcí. Dále zde byl problém v nastavení vlastnosti spojení s FTDI zařízením SetBitMode, kde atribut maska určuje, kterou pozici chceme měřit, ale atribut mod nebyl nastaven podle režimu použitého v mikroprocesoru zařízení. Díky tomu nebylo možno komunikovat s připojeným zařízením, bylo nutné přepnout tento režim na BitBangMode. Po naimplementování funkce pro čtení okamžitých hodnot, byl zjištěn při měření pozic přípravku šum na naměřených napětích. Zde bylo příčinou nastavení vstupního parametru rychlosti měření této funkce na 80 ms. Stačila změna tohoto parametru na 250ms, čímž se prodloužila doba čtení jednotlivých pozic, ale šum byl odstraněn.

Při prvním nasazení aplikace na pracovní PC, bylo nutné nejprve zvýšit nainstalovanou rozhraní .NET Freamworku na verzi 4, protože při implementaci byli použity vlastnosti a jmenné prostory, které v předchozích verzích nejsou dostupné. Např.: generický typ System.Action<T> zapouzdřující metodu bez parametru a navratových hodnot, System.Text.StringBuilder využívaný při získání naměřených hodnot ze zprávy aj.. Po tomto kroku již nebránilo nic ve spuštění aplikace. Po několika přeměřených přípravcích byla odhalena nedostatečná informovanost uživatele v průběhu měření. Do informační oblasti byla přidána informace informující uživatele o prováděných krocích. Zároveň byl přidán do formuláře prvek MessageBox z knihovny System.Windows.Forms, informující uživatele o vzniku náhodných chyb v novém okně s českým textem (odpojení přípravku při testování aj.). Po odladění těchto nedostatků mohla být aplikace předána do užívání.

7 Závěr

Na základě požadavku firmy Elgas s.r.o., je vytvořena aplikace pro kontrolu přípravků PPZ082, které slouží k nastavení převodníků tlaku do plynoměrů, které tato firma vyrábí. Jelikož je to velmi důležitá operace při výrobě převodníku, je potřeba tyto přípravky neustále kontrolovat. Prvním důvodem potřeby této aplikace je nárůst výroby a tím i zvýšení počtu těchto přípravků, a s tím spojených potřebných kontrol. Tato aplikace tuto kontrolu nejen urychlí, ale dovoluje ji provádět zaměstnanci bez potřebných znalostí po krátkém zaškolení. Modul Modbus je naimplementován jako dvě partial třídy, které mají společné vlastnosti, ale oddělují potřebné funkce od logiky protokolu Modbus, což zvyšuje přehlednost řešení. Při návrhu těchto tříd bylo nutné pochopit jak princip tohoto protokolu, časování přenosu, tak i kontrolní výpočty probíhající při přenosu zpráv. Důležité bylo také dodržet dle zadání přenositelnost modulu v případě znovupoužití v jiném projektu, kde by mohly být využity ostatní naprogramované funkce, které v tomto projektu nebyly potřebné. Pro návrh grafického rozhraní byla použita technologie Windows Forms. Zřetel byl brán hlavně na snadné a intuitivní ovládání celé aplikace. Přínosem byla také implementace ukládání záznamů měření, čímž je odstraněna nutnost zapisovat záznamy kontrolovaných přípravků. V budoucnu je možné tuto metodu přepracovat a ukládat tyto data do databáze. Aplikace byla nasazena na pracovním pc, kde byly při testování nečekané chyby odchyceny a po té odstraněny.

7. Seznam použitých zdrojů

[1] **Ing. Ronešová Andrea**. *Přehled protokolu MODBUS*.[online].[cit. 2005-05-26]

URL:<http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf>

[2] **Modbus-IDA**. *MODBUS Protocol Specification* [online].[cit. 2006-12-28]

URL:<http://modbus.com/docs/Modbus_Application_Protocol_V1_1b.pdf>

[3] **Modicon**, *MODBUS Serial Line Protocol and Implementation GuideV1.0* [online].[cit. 2006-12-20].

URL: <http://modbus.com/docs/Modbus_over_serial_line_V1_02.pdf>

[4] **Milan Horkel**, **Jakub Kákona**. *Převodník USB* [online].[cit. 2008-08-20]

URL:<http://www.mlab.cz /miho/2008-08-20/USB232R01B.doc>

[5] **FTDI CHIP, Ltd**. *KATALOGOVÝ LIST: FT232R - ver 2.07* [online].[cit.1.7.2011].

URL:< ww.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>

[6] **FTDI**

URL:<http://ftdichip.com/>

[7] **FTDI**

URL:<http://www.ftdichip.com/Drivers/D2XX.htm/>

[8] **FTDI CHIP**. *Software Application Development D2XX Programmer's Guide Document Reference No.: FT_000071 Version 1.2* .[online].[cit.2011-04-25]. URL:<http://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer's_Guide(FT_000071).pdf/>

[9]**Josef Pirkl**.*Řešené příklady v C# aneb C# skutečně prakticky*.Koop.2005

[10] **PPZ082_04.docx**.*Ing. Jan Věříš* - 4.11.2002

8. Seznam obrázků, tabulek a zdrojových kódů

Obrázky:

Tabulky:

Zdrojové kódy:

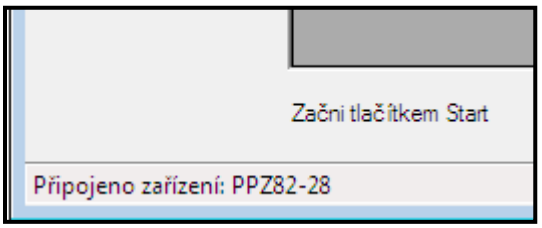
9. Seznam příloh

Příloha	Název
A	Uživatelský manuál aplikace ModbusBakalarka
B	Programová dokumentace
C	CD se zdrojovými kódy, manuálem a instalací aplikace

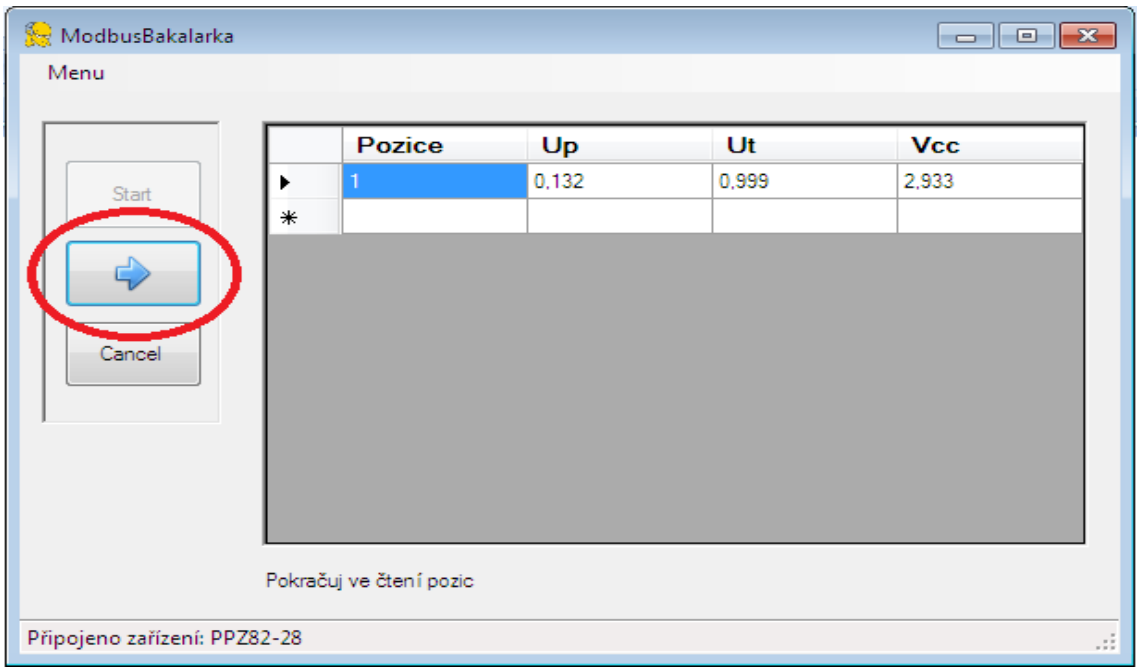
Příloha A

Uživatelský manuál aplikace ModbusBakalarka pro kontrolu přípravku PPZ082

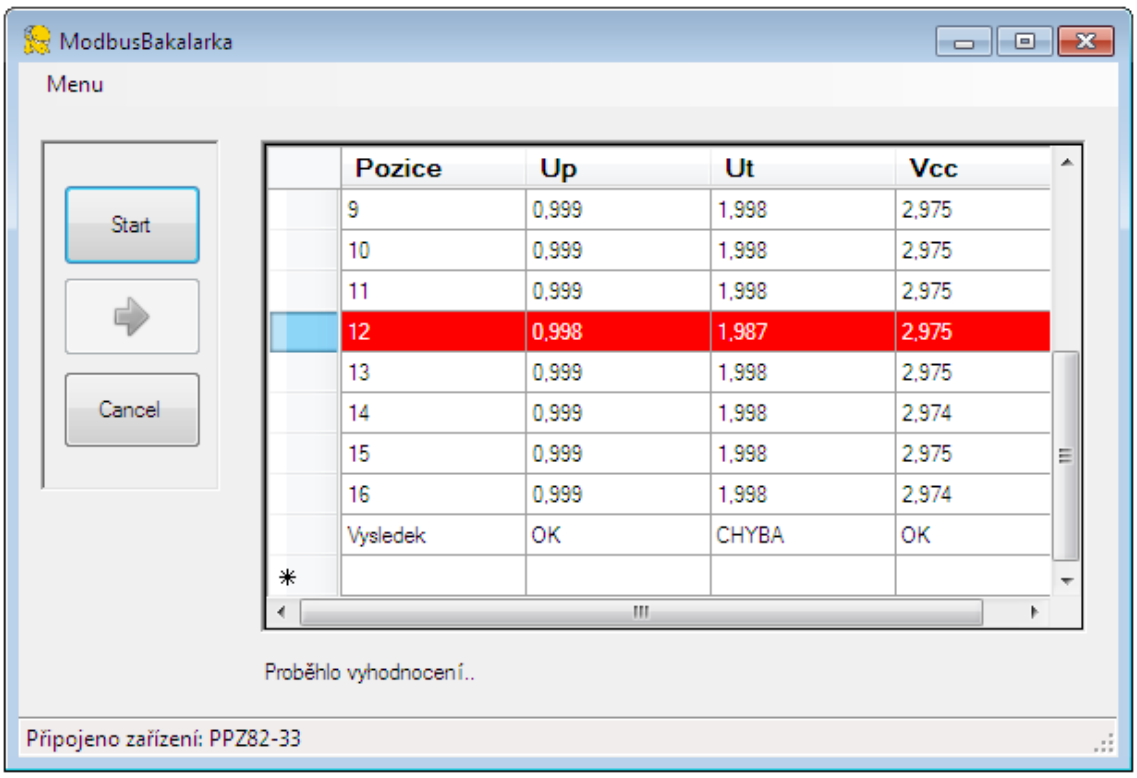
1. Aplikace ModbusBakalarka je určena ke kontrole přípravků PPZ082 dle řízeného dokumentu PPZ082_04.docx.
2. Postup
    a. Připojte přípravek PPZ082 ke zvolenému pracovnímu PC
    Přípravek je možné připojit i po spuštění aplikace, pokud není ještě spuštěn test. Jakmile odpojíte přípravek v průběhu testu, je vyvolána příslušná chyba.
    b. Spusťte program ModbusBakalarka, připojené zařízení je indikováno v dolním panelu formuláře.



    c. Vložte do svorek přípravku na první pozici etalon PPZ 08208
    d. Spusťte test tlačítkem START
    e. Zobrazí se první naměřené hodnoty, tlačítko START přestalo být aktivní
    f. Vyjměte etalon, vložte na další pozici a stiskněte tlačítko se šipkou pro pokračování v měření



g. Postupně pokračujte v kroku f) než proběhne měření všech 16 pozic přípravku
h. Po poslední měřené pozici proběhne vyhodnocení měření všech pozic.

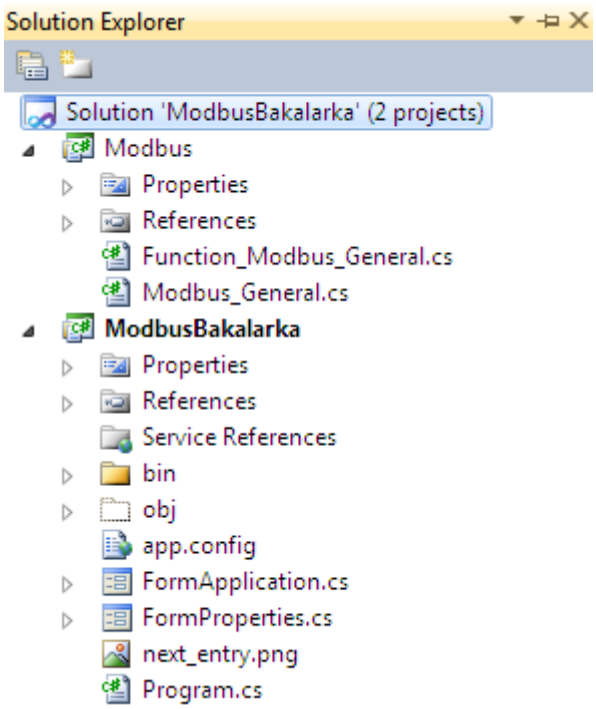


i. Pokud jsou všechna měření napětí Up, Ut, Vcc v pořádku, zobrazí se v řádku výsledek ve všech sloupcích hodnota OK, jakmile ale je na některé z pozic hodnota mimo určenou mez zobrazí se ve výsledku chyba. Zároveň je také červeně obarven řádek pozice, ve které byli naměřeny špatné hodnoty.
j. Připojte další přípravek a pusťte znovu test od bodu c)
k. Tlačítkem CANCEL můžete kdykoli zrušit průběh měření, a dostanete se do úvodního stavu
l. Po každém proběhlém měření je uložen výsledek do souboru vysledek.txt v adresáři aplikace

Příloha B

Programová dokumentace aplikace ModbusBakalarka

Solution:



Modul Modbus se skládá ze dvou částečných (partial) tříd:1)ModbusGeneral a 2)ModbusFunction .Tyto třídy oddělují funkční metody (add.2) od základních metod pro práci s protokolem Modbus(add.1). Do tohoto modulu je přidána reference na knihovnu FT2XX.NET.dll, která nám zpřístupňuje funkce potřebné pro práci s FTDI zařízením Modul Modbus je potom přidán jako reference do formuláře ApplicationForm, který propojuje logiku aplikace s uživatelskou částí aplikace.

Třída ModbusGeneral

`void conectSerial()`
Bezparametrická, bez návratová metoda, která navazuje spojení s FTDI zařízením, nejprve načte seznam připojených zařízení do pole `ftdiDeviceList` a po té vytvoří připojení s vybraným FTDI zařízením ze seznamu. Defaultně nastaveno na první zařízení v seznamu:
`myFtdiDevice.OpenBySerialNumber(ftdiDeviceList[0].SerialNumber)`

`void CloseSerial()`
Bez návratová metoda ukončující spojení se zařízením.

`Byte[] buildMessage(byte address, byte function, byte start, byte count, byte[] data)`
Tato metoda sestavuje zprávu odesílanou do FTDI zařízení. Vstupem jsou jednotlivé bajty, z kterých se skládá zpráva a jejichž pozice je pevně dána. Výstup tvoří bajtové pole tvořící vstupní proměnné + vypočítaný kontrolní součet CRC.

`Byte[] CRC (byte[] dats)`
Metoda, která vrací kontrolní součet poslané/příjmuté zprávy. Vstupní proměnnou je bajtové pole jakékoliv velikosti. Výstupem je dvou bajtové pole s vypočítaným kontrolním součtem.

`void response()`
Tato metoda získá odpověď z FTDI zařízení na předcházející dotaz a uloží ji do lokální proměnné finalResponse. Metoda čeká v cyklu na bajty vrácené z FTDI zařízení ke čtení. Při čtení kontroluje mezi znakové časy a podle toho vyhodnocuje konec zprávy či chybnou zprávu. Je zde také volána metoda `control` kontrolující přijatou zprávu.

`void Control(byte[] answer)`
Slouží ke kontrole přijatých zpráv, a v případě odchycení chyby tuto chybu identifikuje. Vstupní parametrem je bajtové pole, které má být zkontrolováno.

`toValues(byte[] fine)`
Zajišťuje získání naměřených hodnot z přijaté zprávy. Tato metoda je volána v metodě `response()` pro získání odpovědi na definovaný požadavek. Vstupní proměnná je zde bajtové pole přijeté zprávy. Z této zprávy se nejprve oddělí adresa a funkce na začátku zprávy a potom kontrolní součet CRC. Zbytek zprávy se rozdělí pomocí algoritmu na tři části měřených hodnot, v kterých jsou procházeny jednotlivá bajtová pole a ukládány do proměnné `sb` typu `System.Text.StringBuilder.` Ta je potom upravena metodami `Parse (Int32.Parse) a BitConverter (System.BitConverter)` na desetiné tvary. Nakonec jsou tyto hodnoty zaokrouhleny.

Metody ModbusFunction
Implementace těchto metod je velice podobná, liší se pouze vstupními parametry a inicializací metody BuildMessage.

`void readInstantValues (byte address, byte data)`
Funkce čtení okamžitých hodnot, které získá po dotazu na měření mikroprocesoru v připojeném zařízení. Vstupním parametrem je bajt adresy zařízení a bajt nastavující rychlost měření, kterou nastavujeme prodlevu mezi dotazem a odpovědí měření. V této metodě je nejprve sestavena zpráva pomocí metody `BuildMessage.`

Po té je v novém vlákně zavolána metoda `Write` z knihovny `FT2XX.NET`, zajišťující zápis dat do FTDI zařízení. Je zde vypočítána hodnota 3.5 znaku času, a toto vlákno je vynuceno počkat tento čas a vytvořit tak potřebnou mezi znakovou mezeru. Hlavní vlákno je po té pozastaveno a čeká na odpověď ze zařízení zavoláním metody `response`.

`void readEEPROM (byte address,byte cont, int point)`
Tato metoda čte jednu stránku (32bajtů) z připojené paměti EEPROM po dotazu do mikroprocesoru. Vstupním parametrem je bajt adresy zařízení, atribut `cont` je zde podfunkce zadaná do dotazu a atribut `point` je ukazatel na adresu prvního čteného bajtu z připojené EEPROM.

`void zápisEEPROM (byte address, byte cont, byte[] point)`
Po jejím zavolání mikroprocesor zapíše jednu stránku do připojené paměti EEPROM. Vstupním parametrem je bajt adresy zařízení, atribut `cont` je zde podfunkce zadaná do dotazu a bajtové pole `point` pole určené k zápisu do připojené EEPROM.

`void read_Holding_Registers (byte address, byte startAddress, byte count)`
Tato metoda slouží ke čtení obsahu vnitřních registrů zařízení. Vstupním parametrem je bajt adresy zařízení, bajt adresy prvního čteného registru a atribut `count` je zde počet registrů, které mají být přečteny.

`void write_Single_Coil(byte address, byte values)`
Nastavuje jeden výstup spínaného zařízení do stavu ON nebo OFF. Vstupním parametrem je bajt adresy zařízení, bajt `values` je hodnota stavu, do kterého se má zařízení uvést.

`void write_Multiple_Registers(byte address, byte startAddress, byte count, byte[] data)`
Tato metoda slouží k zápisu bloku až 120 registrů. Vstupním parametrem je bajt adresy zařízení, bajt `startAddress` je adresa počátečního registru, `count` je zde počet registrů určených k zápisu a bajtové pole `data` jsou hodnoty, které se mají zapsat.

`void report_Slave_ID(byte address)`
Slouží k zjištění typu, stavu a dalších informací o zařízení. Konkrétní obsah odpovědi závisí na typu připojeného zařízení. Vstupním parametrem je bajt adresy zařízení.

Hlavní metody ApplicationForm

`void defaultStartup()`
Tato metoda slouží k základnímu nastavení parametrů FTDI připojení jako je: čas zápisu/čtení, přenosová rychlost, pozice čtení. Přistupuje se zde k proměnným knihovny FTDI např:

```
SetDataCharacteristics(FTD2XX_NET.FTDI.FT_DATA_BITS.FT_BITS_8,
                       FTD2XX_NET.FTDI.FT_STOP_BITS.FT_STOP_BITS_1,
                       FTD2XX_NET.FTDI.FT_PARITY.FT_PARITY_NONE)
```

`void connect()`
Metoda obsluhující připojení FTDI zařízení. Tato metoda je volána po spuštění aplikace a pokud nenalezne žádné zařízení je spuštěna ve vlákně na pozadí, kde čeká na připojení zařízení. Na začátku metody probíhá kontrola, zda zařízení již není připojeno. Jeli nalezeno zařízení, je zavolána metoda `connectSerial()`, která dané zařízení připojí. Pokud zařízení připojí, umožní uživateli přístup k prkům formuláře, které spouští test přípravku.

`void test()`
V této funkci probíhá vlastní měření pozic přípravku PPZ082. Jou zde vytvořeny nejprve potřebné řádky `DataTable (knihovna System.Data)` pro naměřené hodnoty. Potom je nastavena měřená pozice přípravku proměnnou `point (Int32)`, která je automaticky měněna podle měřené pozice. Po nastavení této pozice je zavolána metoda `ReadInstantValues` pro čtení okamžitých hodnot převodníku. Pokud vše proběhne v pořádku je zavolána metoda pro vyhodnocení zprávy `toValues(byte[] fine)`, pro získání hodnot z výsledku v desetinném tvaru. Tyto hodnoty jsou uloženy do vytvořených řádků DataTable a ta je přidána do zobrazovacího prvku `DataGridView (knihovna System.Data)` jako datový zdroj, který změřená data prezentuje uživateli.

`void writeTxt(string up, string ut, string vcc)`
Ta má za úkol, zapisovat výsledek testu do textového souboru “výsledek.txt“. Vstupní proměnné jsou řetězcové vyjádření naměřených výsledků, které jsou uloženy do proměnné `System.IO.StreamWriter` určeného pro zápis znaků do proudu. Dále je zde uložen datum a seriové číslo měřeného přípravku `ftdiDeviceList[0].SerialNumber.`